**UVOD U ARDUINO**

**Arduino** je fizičko-računarska platforma ([razvojni sistem](https://sr.wikipedia.org/w/index.php?title=Razvojni_sistem&action=edit&redlink=1)) [otvorenog koda](https://sr.wikipedia.org/wiki/%D0%9E%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%B8_%D0%BA%D0%BE%D0%B4). [Hardver](https://sr.wikipedia.org/wiki/%D0%A5%D0%B0%D1%80%D0%B4%D0%B2%D0%B5%D1%80) se sastoji od jednostavnog otvorenog hardverskog dizajna Arduino ploče sa Atmel AVR [procesorom](https://sr.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80) i pratećim ulazno-izlaznim elementima, tačnije, na sebi poseduje mikrokontroler. [Softver](https://sr.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D0%B2%D0%B5%D1%80) se sastoji od razvojnog okruženja koje čine standardni [kompajler](https://sr.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D1%98%D0%BB%D0%B5%D1%80) i [bootloader](https://sr.wikipedia.org/wiki/Bootloader) koji se nalazi na samoj ploči.

Arduino hardver se programira koristeći programski jezik zasnovan na Wiring jeziku (sintaksa i biblioteke). U osnovi je sličan C++ programskom jeziku sa izvesnim pojednostavljenjima i izmenama. [Integrisano razvojno okruženje](https://sr.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%81%D0%B0%D0%BD%D0%BE_%D1%80%D0%B0%D0%B7%D0%B2%D0%BE%D1%98%D0%BD%D0%BE_%D0%BE%D0%BA%D1%80%D1%83%D0%B6%D0%B5%D1%9A%D0%B5) je zasnovano na Processing-u.  
Tajna velike popularnosti *Arduino* platforme leži u činjenici da ona po svojoj koncepciji uveliko pojednostavljuje tradicionalni proces konstruisanja elektroničkih sklopova. Zahvaljujući mnoštvu gotovih modula koji se jednostavno dodaju u projekat, stvoreni su uslovi da i apsolutni početnici u svetu elektronike mogu u roku od nekoliko minuta formirati sklopove koji obavljaju dovoljno kompleksne zadatke. Naravno, nećemo daleko odmaći bez poznavanja osnovnih pojmova kao što je Omov zakon ili kakva je funkcija otpornika, kondenzatora i tranzistora, ali rad sa modulima omogućava početnicima da na najbezbolniji mogući način savladaju filozofiju funkcionisanja strujnih kola. Ali to nije sve. Uporedo sa jednostavnim korišćenjem hardvera, kreatori *Arduino* platforme su učinili možda i još važniju stvar sa uvođenjem mogućnosti jednostavnog programiranja naših hardverskih uređaja.

**Arduino IDE**

|  |  |
| --- | --- |
| **[https://www.sk.rs/2016/11/sklp05b.jpg](https://www.sk.rs/2016/11/sklp05b.velika.jpg)** |  |

Istorija celog projekta *Arduino* vezana je za programski alat pod nazivom *Processing* od kojega je preuzeo mnoštvo stvari. *Processing* je zanimljiva platforma koja omogućuje pisanje aplikacija na pojednostavljenoj varijanti jezika *Java*. Reč je o jednom od najefektivnijih načina za programiranje na platformi *Android* (vlasnicima telefona toplo savetujemo da postave program *APDE),* a podržane su i sve popularne desktop platforme. Dodavanjem biblioteke funkcija je moguće i programiranje *Arduino* uređaja direktno iz ove platforme. *Arduino* je od *Processinga* preuzeo razvojno okruženje koje je inače napisano na jeziku *Java*, čemu dugujemo njegovu portabilnost. Tu imamo dobru i lošu vest. Dobra je da je IDE izuzetno jednostavan za korišćenje i da će se u njemu bez problema snaći svaki početnik. Loša vest je ta da će iskusniji korisnici pronaći mnoštvo ograničenja za rad sa složenijim projektima. Od naprednijih opcija, tu su sintaksno bojenje teksta (doduše, bez mogućnosti izbora sopstvenih boja), automatsko formatiranje koda (opcija u meniju *Tools*) i skrivanje koda u okviru funkcija. Nažalost, nisu podržane neke danas standardne mogućnosti, kao što je prikazivanja sintaksne greške prilikom editovanja ili autokompletiranje koda. U jednom od narednih brojeva osvrnućemo se na radna okruženja koja u sebi sadrže sve ono što *Arduino* IDE iz nekih razloga nije podržao, ali za sada nam je on više nego dovoljan.

Kada završimo sa pisanjem skeča (engl. *scatch,* naziv programa pisanih za *Arduino*), potrebno je da proverimo da li sintaksa ima greške uz pomoć opcije *Verify*. Ukoliko nema grešaka, na dnu prozora će se pokazati podaci o zauzeću memorije za pohranjivanje programa i koliko je bajtova otišlo na podatke u RAM. U prošlom broju smo pisali da *Arduino Uno*ima 32 kilobajta programske memorije i dva kilobajta SRAM, tako da je potrebno ekonomično upravljanje sa ovim resursom. Opcija *Upload* kompajlira kod i prebacuje ga na mikrokontroler. Da bismo skeč pokrenuli na našem uređaju, potrebno je da prethodno iz menija *Tools: Board* izaberemo model koji trenutno koristimo. Osim toga je iz menija *Tools: Port* potrebno izabrati serijski port koji je dodeljen našem USB adapteru preko koga vršimo programiranje *Arduina*.

Već smo pominjali da je za ovu platformu napisan veliki broj programskih biblioteka koje uveliko olakšavaju korišćenje svakojakih hardverskih modula koje možemo uključiti u naše projekte. Umesto da se sami mučimo sa realizacijom neke funkcije (recimo, želimo da crtamo po displeju) jednostavno se poslužimo sa gotovom bibliotekom koju su neki dobri ljudi napisali pre nas. Ukoliko želimo da dodamo biblioteku koja nam je potrebna, to je najlakše obaviti preko menija *Sketch: Import Library*. U istom meniju ćemo videti sve biblioteke koje već postoje na našem sistemu. Izborom neke od njih u program će biti automatski dodata sva neophodna zaglavlja preko *#include <>* direktiva.

**Iz tona C**

Osnovni programski jezik koji se koristi na platformi *Arduino* je svojevrsna pojednostavljena varijanta jezika *C.* Tačnije, iz jezika su sklonjene neke kompleksnije stvari i uvedena je *C/C++* biblioteka zvana *Wiring*, koja je preuzeta iz pomenute platforme *Processing*. Upravo iz činjenice da je *Processing* pisan za jezik *Java*, a *Wiring* za *C/C++* proizilaze i nevelike razlike u njihovoj sintaksi.

Iako *C* važi za jedan od težih jezika za savladavanje, ovde je to teško primetiti. Kostur svakog *Arduino* programa kreiranog uz pomoć razvojnog okruženja *Arduino* IDE izgleda ovako:

void setup() {

  // put your setup code here, to run once:

}

void loop() {

  // put your main code here, to run repeatedly:

}

Kao što vidimo, reč je o dve funkcije od kojih je prva (*setup*) zadužena za inicijalizaciju parametara programa i izvodi se samo jednom na početku, dok funkcija *loop* predstavlja beskonačnu petlju koja se ponavlja tokom rada programa. Pošto se radi o varijanti jezika *C,* mnogi će se upitati šta je sa nezaobilaznom funkcijom *main*()? Da bi pojednostavili rad, autori su rešili da je prosto sakriju, tako da ona i dalje obavlja svoj posao iza scene. U stvarnosti, osnovni skelet *Arduino* skeča izgleda ovako:

int main() {

init();           //inicijalizuj USB

setup();          //početne postavke

while(1){loop();} //beskonačna petlja

}

Funkcija *init()* reguliše neke parametre vezane za USB port. Sledi pozivanje funkcije *setup()* za koju smo pisali da je namenjena izvođenju operacija koje se obavljaju samo jednom na početku programa. Posle toga ulazimo u beskonačnu petlju koja će stalno pozivati funkciju *loop()*sve do prestanka rada, a u našem slučaju to znači do prekida dotoka električne energije ili pritiska na reset taster.

|  |
| --- |
| **[https://www.sk.rs/2016/11/sklp05c.gif](https://www.sk.rs/2016/11/sklp05c.velika.gif)** |

Zanimljivo je da skoro svi udžbenici na ovu temu za prvo upoznavanje sa kodom navode demonstracioni primer pod nazivom *Blink*. Razlog za to je svakako taj što svi modeli *Arduina* na sebi imaju malu LED lampicu koja je fizički povezana sa digitalnim izlazom 13 i pokazuje njegovo trenutno stanje. Ukoliko postavimo LED sa anodom utaknutom u pin 13 i katodom u pin GND, videćemo isti efekat. Dakle, moguće je demonstrirati rad uređaja bez ikakve dodatne opreme.

void setup() {

  pinMode(13, OUTPUT);

}

void loop() {

  digitalWrite(13, HIGH);

  delay(1000);

  digitalWrite(13, LOW);

  delay(1000);

}

Unutar funkcije *setup(),* digitalnom U/I pinu broj 13 funkcijom *pinMode()* dodeljujemo funkciju izlaza. To je potrebno zbog toga što U/I pinovi mogu obavljati kako funkciju ulaza, tako i izlaza. Iako su svi U/I pinovi inicijalno postavljeni u režim rada INPUT, moguće ih je koristiti bez pozivanja funkcije *pinmode()* ukoliko preko njih očekujemo prijem podataka. Ipak, toplo se savetuje da se u cilju izbegavanja nesporazuma ta funkcija uvek eksplicitno koristi.

Unutar beskonačne petlje *loop*(), pomoću funkcije *digitalWrite*() na pinu 13 postavljamo stanje takozvane logičke jedinice uz pomoć konstante HIGH, što dovodi do uključi-vanja LED lampice. Logička jedinica i nula predstavljaju opsege gornjeg i donjeg napona sa kojim funkcioniše neka komponenta. Generalno, kod modela UNO i MEGA2560 ona predstavlja napon od dva do pet volti, dok logička nula predstavlja napon u opsegu od nula do jedan volt. Kod modela koji rade na 3,3 volta sa HIGH predstavljamo naponsko stanje od dva do 3,3 volta, dok je LOW jednako naponu u opsegu od nula do 0,8 volta. U praksi su ovi opsezi nešto uži, recimo, u slučaju TTL kola gornji napon je u oblasti 2,7-5 volti, dok je donji 0-0,4 volti. Umesto konstanti HIGH i LOW mogli smo da koristimo brojeve 0 i 1 kako bismo označili u kakvom se logičkom stanju trenutno nalazi ukazani pin, ali je ipak elegantnije raditi sa njima.

Nakon što smo uključil LED lampicu, mikrokontroler izvršava funkciju *delay*(1000) koja pravi pauzu od 1000 milisekundi, odnosno jedne sekunde. Sledeći korak je izvođenje funkcije *digitalWrite()* sa kojom na digitalni izlaz 13 šaljemo stanje logičke nule, koje izaziva isključenje lampice i to u intervalu od 1000 milisekundi nakon čega nas beskonačna petlja ponovo vraća na početak. Rezultat rada programa je smenjivanje perioda kada dioda svetli i kada ne emituje svetlost.

|  |
| --- |
| **[https://www.sk.rs/2016/11/sklp05d.gif](https://www.sk.rs/2016/11/sklp05d.velika.gif)** |

Još jedan vrlo jednostavan primer koji demonstrira funkcionisanje PWM tehnike digitalno-analogne konverzije od nas zahteva da na uređaj priključimo LE diodu i otpornik od 220 oma.

Na anodu LE diode (poznaje se po dužoj nožici i maloj izbočini na plastici) dovodimo signal sa pina 10 preko otpornika od 220 oma (zavisi od vrste diode koja se koristi, u najvećem broju slučajeva će poslužiti otpornici od 160-600 oma) koji je štiti od pregorevanja. Na katodu (kraća nožica) dovodimo uzemljenje sa pina GND (na ilustraciji uzemljenje prvo dovodimo na liniju uzemljenja prototipske ploče pa odatle na katodu). Pin broj 10 *Arduina* *Uno* (kao i pinovi 3, 5, 6, 9 i 11) ima funkciju pretvaranja digitalnog signala u analogni ekvivalent tako što emituje 256 različitih nivoa napona u opsegu nula do pet volti. Pa da vidimo kako se to ostvaruje programski:

int svetlost=0;  // intenzitet LED svetla

void setup() {

pinMode(10, OUTPUT); // pin 10 je PWM izlaz

}

void loop() {

  for (svetlost=0;svetlost<255;svetlost++)

    {

    analogWrite(10, svetlost);

    delay(10); // kratka pauza

    }

  for (svetlost=255;svetlost>0;svetlost--)

    {

    analogWrite(9, svetlost);

    delay(10); // kratka pauza

    }

delay(2000); // duža pauza

}

Nivo osvetljenosti LE diode koji želimo da zadamo se nalazi u varijabli *svetlost* koju preko *for-next* petlje šaljemo na pin 10, povećavajući je do maksimalne vrednosti 255, da bismo zatim na isti način tu vrednost smanjivali do nule. Između svakog koraka pravimo pauzu od 20 milisekundi kako bi bio vidljiv efekat postepenog uključivanja i isključivanja lampice. Iza svakog ciklusa uključivanja i isključivanja imamo pauzu od dve sekunde. Primer je suštinski dosta sličan prethodnom sa jednom razlikom da ovde umesto funkcije *digitalWrite* koristimo *analogWrite*, pošto ova prva ima mogućnost prikazivanja samo dva stanja: uključeno i isključeno. Cela priča vezana za *Arduino* programiranje se uglavnom vrti oko čitanja i upisivanja analognih i digitalnih vrednosti, a ona je zahvaljujući konceptu platforme *Wiring* izvedena na jednostavan i razumljiv način.

Od ostalih stvari koje inicijalno dolaze u ovoj varijanti jezika C/C++, na raspolaganju su nam više-manje standardne funkcije za konverziju među različitim tipovima podataka *(byte(), char(), int(), word(), long(), float()),* matematičke funkcije *(min(), max(), abs(), sqrt()* i tako dalje), trigonometrijske *(sin(), cos(), tan()),*vremenske *(delay(), millis(), micros())*za generisanje slučajnih brojeva *(random(), randomseed())...* Što se tiče manipulacije sa tekstom, izvorno je podržan rad sa baznim nizovima alfanumeričkih karaktera, ali nam je na raspolaganju i klasa *String* koja sa sobom donosi niz naprednih metoda za rad sa tekstom. Sa jedne strane, rad sa tipom podataka *String* i metodima istoimene klase pruža programeru moćno sredstvo za obradu ove vrste podataka, dok sa druge strane generiše dosta više koda, pa je potreban oprez u situacijama kada nemamo mnogo resursa na raspolaganju.

Ovaj pregled nije rađen sa namenom da bude udžbenik programiranja na platformi *Arduino*, već da posluži kao demonstracija jednostavnosti implementacije jednog dosta kompleksnog jezika kao što je *C.* Postoji mnogo udžbenika i tutorijala na temu savladavanja tog programskog jezika, pa vam, ukoliko ne poznajete njegovu sintaksu, toplo savetujemo da pročitate neki od njih. Svi oni koji poseduju veštinu rada sa nekim od programskih jezika izvedenih iz C (*Java*, *JavaScript*, *C#*...) sasvim sigurno će u vrlo kratkom roku biti u stanju da pišu skečeve za *Arduino*.

**Hardver**

Arduino ploču čine 8-bitni Atmel AVR [mikrokontroler](https://sr.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80) sa pripadajućim komponentama koje omogućavaju programiranje i povezivanje sa drugom elektronikom. Bitan aspekt Arduino projekta je standardizovan raspored konektora koji omogućava lako povezivanje sa dodatnim modulima, poznatijim kao štitovi. Ove dodatne module, štitove, poizvode razni proizvođači širom sveta. Zvanične Arduino ploče uglavnom koriste megaAvr seriju čipova, konkretno ATmega8, ATmega168, ATmega328, ATmega1280 i ATmega2560. Većina ploča poseduje 5V linearni naponski regulator i 16 MHz kristalni oscilator (ili keramički rezonator u nekim verzijama). Arduino mikrokontroleri se isporučuju sa programiranim [bootloader](https://sr.wikipedia.org/wiki/Bootloader)-om koji pojednostavljuje postupak prebacivanja prevedenog koda u [fleš memoriju](https://sr.wikipedia.org/wiki/%D0%A4%D0%BB%D0%B5%D1%88_%D0%BC%D0%B5%D0%BC%D0%BE%D1%80%D0%B8%D1%98%D0%B0) na čipu. Drugi mikrokontroleri obično zahtevaju zaseban [programator](https://sr.wikipedia.org/w/index.php?title=Programator&action=edit&redlink=1).

**Softver**

Arduino integrisano razvojno okruženje je aplikacija napisana u Java programskom jeziku. Kreirano je tako da uvede u programiranje učenike, studente i ostale početnike koji nisu upoznati sa načinom razvoja softvera. Sastoji se od uređivača koda sa mogućnostima kao što su označavanje koda, uparivanje zagrada, automatsko uvlačenje linija. Ovaj uređivač može da prevede kôd a zatim ga i prebaci u čip jednom komandom. U ovom slučaju nije potrebno podašavati parametre prevođenja koda ili pokretati programe iz komandne linije.

Arduino integrisano razvojno okruženje dolazi sa C/C++ bibliotekom zvanom "Wiring" koja čini uobičajene ulazno-izlazne operacije veoma jednostavnim. Arduino programi se pisu u C/C++ programskom jeziku, mada korisnici moraju da definišu samo dve funkcije kako bi napravili izvršni program. Te funkcije su:

* setup() - funkcija koja se izvršava jednom na početku i služi za početna podešavanja
* loop() - funkcija koja se izvršava u petlji sve vreme dok se ne isključi ploča

Tipičan prvi program jednostavno pali i gasi [LED](https://sr.wikipedia.org/wiki/%D0%A1%D0%B2%D0%B5%D1%82%D0%BB%D0%B5%D1%9B%D0%B0_%D0%B4%D0%B8%D0%BE%D0%B4%D0%B0) diodu. U Arduino okruženju, korisnik bi mogao da napše ovakav program:

#define LED\_PIN 13

void setup () {

pinMode (LED\_PIN, OUTPUT); *// definiši pin 13 kao digitalni izlaz*

}

void loop () {

digitalWrite (LED\_PIN, HIGH); *// uključi LED*

delay (1000); *// sačekaj jedan sekund (1000 milisekundi)*

digitalWrite (LED\_PIN, LOW); *// isključi LED*

delay (1000); *// sačekaj jedan sekund*

}

Da bi ovaj kôd ispravno radio, anoda LED diode mora biti povezana na pin 13 a katoda LED diode na uzemljenje (u ovom slučaju negativni deo napajanja). Ovaj primer od strane standardnog C++ prevodioca ne bi bio viđen kao ispravan program, međutim kada korisnik izda komandu za prevođenje, ovom kodu se dodaje izvesno zaglavlje i jedna jednostavna main() funkcija čime on postaje ispravan.

Arduino razvojno okruženje koristi [GNU toolchain](https://sr.wikipedia.org/w/index.php?title=GNU_toolchain&action=edit&redlink=1) i [AVR Libc](https://sr.wikipedia.org/w/index.php?title=AVR_Libc&action=edit&redlink=1) za prevođenje programa.